Amendments to the Claims:

This listing of claims will replace all prior versions, and listings, of claims in the application:

1. (original) A method for reorganizing rows from a partitioned database table, the partitioned database table including a plurality of populated partitions, comprising the steps of:
    a. organizing rows in each of the populated partitions in accordance with a first value associated with each row;
    b. creating a file context for each partition of a subset of the populated partitions, each file context storing at least location data for a row in the partition and the first value associated with the row;
    c. merging rows from the subset of partitions into a single first-merge partition in order of the first value associated with each row;
    d. repeating steps b through c until the subsets have included all populated partitions.

2. (original) The method of claim 1, further comprising the step of:
    e. comparing a specified grouping limit to the number of first-merge partitions and merging the first-merge partitions if the specified grouping limit is less than the number.

3. (original) The method of claim 1, wherein the location data for a row is the location of a block of rows that includes the row.

4. (original) The method of claim 1, wherein steps a through c are performed on rows in a single data-storage facility.

5. (original) The method of claim 1, wherein the file contexts are stored in memory.

6. (original) The method of claim 1, wherein the rows of the first-merge partitions are stored separately from the rows of the populated partitions of the partitioned database table.

7. (original) The method of claim 1, further comprising the steps of:
    a'. determining whether rows from a partitioned primary index table are being spooled;

a". determining whether a subsequent operation requires the spooled rows to be ordered in accordance with the first value associated with each row; and

a'". performing steps b through d only if both determinations, a' and a", are true.

8. (previously presented) The method of claim 11, wherein the specified grouping limit is 1.

9. (original) The method of claim 8, wherein first-merge partitions and spool-merge partitions are contained in different subtables of a spool.

10. (original) The method of claim 8, wherein step j includes merging rows from the subset of spool-merge partitions, each located in a first subtable of a spool, into a new spool-merge partition, located in a second subtable of the spool.

11. (original) The method of claim 1, further comprising the steps of:

e. creating a file context for each first-merge partition of a subset of the first-merge partitions, each file context storing at least location data for a row in the partition and the first value associated with the row;

f. merging rows from the subset of first-merge partitions into a spool-merge partition in order of the first value associated with each row;

g. repeating steps e and f until the subsets have included all first-merge partitions;

h. bypassing steps i through k if a specified grouping limit is at least equal to the number of spool-merge partitions;

i. creating a file context for each spool-merge partition of a subset of the spool-merge partitions, each file context storing at least location data for a row in the partition and the first value associated with the row;

j. merging rows from the subset of spool-merge partitions into a new spool-merge partition in order of the first value associated with each row;

k. repeating steps i and j until the specified grouping limit is at least equal to the number of remaining spool-merge partitions.

12. (original) The method of claim 1, wherein the subsets of partitions contain no more than a specified number of populated partitions and the specified number is determined by memory usage.

13. (original) The method of claim 1, further comprising the step of:

a'. calculating the cost of reorganizing rows from a partitioned database table using the equation cost = (r1 + w) + ((r2 +w) * (ceiling($\log_m$p)-1)), wherein r1 is the cost to read and qualify rows in non-eliminated partitions, w is the cost to write qualifying rows to a spool, r2 is the cost to read the rows in the spool, m is the number of partitions in a subset, p is the number of populated partitions in the table, and ceiling returns an integral argument rounding up.

14. (original) The method of claim 1, wherein the reorganization is conducted in response to a query having conditions and the step of merging rows includes eliminating rows that do not satisfy the query conditions.

15. (original) The method of claim 1, wherein the first subset of the populated partitions includes all the populated partitions and steps b and c are not repeated.

16. (original) The method of claim 1, wherein the first value is the result of a hash function applied to one or more values in one or more columns of the associated row.

17. (currently amended) A database system for reorganizing rows from a partitioned database table, the partitioned database table including a plurality of populated partitions, the system comprising:

one or more nodes;

a plurality of CPUs, each of the one or more nodes providing access to one or more CPUs;

a plurality of virtual processes, each of the one or more CPUs providing access to one or more virtual processes;

each virtual process configured to manage data, including rows from the partitioned database table, stored in one of a plurality of data-storage facilities;

a partition merging component employing at least one of the plurality of virtual processes and configured to reorganize rows from the partitioned database table in each data-storage facility by:

a. organizing rows in each of the populated partitions in accordance with a first value associated with each row;

b. creating a file context for each partition of a subset of the populated partitions, each file context storing at least location data for a row in the partition and the first value associated with the row;

c. merging rows from the subset of partitions into a single first-merge partition in order of the first value associated with each row;

d. repeating steps b through c until the subsets have included all populated partitions.

18. (original) The database system of claim 17, wherein the partition merging component reorganizes rows by:

e. comparing a specified grouping limit to the number of first-merge partitions and merging the first-merge partitions if the specified grouping limit is less than the number.

19. (original) The database system of claim 17, wherein the location data for a row is the location of a block of rows that includes the row.

20. (original) The database system of claim 17, wherein the file contexts are stored in memory.

21. (original) The database system of claim 17, wherein the rows of the first-merge partitions are stored separately from the rows of the populated partitions of the partitioned database table.

22. (original) The database system of claim 17, wherein the partition merging component reorganizes rows by:

a'. determining whether rows from a partitioned primary index table are being spooled;

a". determining whether a subsequent operation requires the spooled rows to be ordered in accordance with the first value associated with each row; and

a'". performing steps b through d only if both determinations, a' and a", are true.

23. (previously presented) The database system of claim 26, wherein the specified grouping limit is 1.

24. (original) The database system of claim 23, wherein first-merge partitions and spool-merge partitions are contained in different subtables of a spool.

25. (original) The database system of claim 23, wherein step j includes merging rows from the subset of spool-merge partitions, each located in a first subtable of a spool, into a new spool-merge partition, located in a second subtable of the spool.

26. (original) The database system of claim 17, wherein the partition merging component reorganizes rows by:

e. creating a file context for each first-merge partition of a subset of the first-merge partitions, each file context storing at least location data for a row in the partition and the first value associated with the row;

f. merging rows from the subset of first-merge partitions into a spool-merge partition in order of the first value associated with each row;

g. repeating steps e and f until the subsets have included all first-merge partitions;

h. bypassing steps i through k if a specified grouping limit is at least equal to the number of spool-merge partitions;

i. creating a file context for each spool-merge partition of a subset of the spool-merge partitions, each file context storing at least location data for a row in the partition and the first value associated with the row;

j. merging rows from the subset of spool-merge partitions into a new spool-merge partition in order of the first value associated with each row;

k. repeating steps i and j until the specified grouping limit is at least equal to the number of remaining spool-merge partitions.

27. (original) The database system of claim 17, wherein the subsets of partitions contain no more than a specified number of populated partitions and the specified number is determined by memory usage.

28. (original) The database system of claim 17, wherein the partition merging component reorganizes rows by:

a'. calculating the cost of reorganizing rows from a partitioned database table using the equation cost = (r1 + w) + ((r2 +w) * (ceiling($\log_m$p)-1)), wherein r1 is the cost to read and qualify rows in non-eliminated partitions, w is the cost to write qualifying rows to a spool, r2 is the cost to read the rows in the spool, m is the number of partitions in a subset, p is the number of populated partitions in the table, and ceiling returns an integral argument rounding up.

29. (original) The database system of claim 17, wherein the reorganization is conducted in response to a query having conditions and the step of merging rows includes eliminating rows that do not satisfy the query conditions.

30. (original) The database system of claim 17, wherein the first subset of the populated partitions includes all the populated partitions and steps b and c are not repeated.

31. (original) The database system of claim 17, wherein the first value is the result of a hash function applied to one or more values in one or more columns of the associated row.

32. (original) A computer program, stored in a tangible medium, for reorganizing rows from a partitioned database table, the program comprising executable instructions that cause a computer to:

a. organize rows in each of the populated partitions in accordance with a first value associated with each row;

b. create a file context for each partition of a subset of the populated partitions, each file context storing at least location data for a row in the partition and the first value associated with the row;

c. merge rows from the subset of partitions into a single first-merge partition in order of the first value associated with each row;

d. repeat steps b through c until the subsets have included all populated partitions.

33. (original) The computer program of claim 32, wherein the executable instructions cause the computer to:

e. compare a specified grouping limit to the number of first-merge partitions and merging the first-merge partitions if the specified grouping limit is less than the number.

34. (original) The computer program of claim 32, wherein the location data for a row is the location of a block of rows that includes the row.

35. (original) The computer program of claim 32, wherein steps a through c are performed on rows in a single data-storage facility.

36. (original) The computer program of claim 32, wherein the file contexts are stored in memory.

37. (original) The computer program of claim 32, wherein the rows of the first-merge partitions are stored separately from the rows of the populated partitions of the partitioned database table.

38. (original) The computer program of claim 32, wherein the executable instructions cause the computer to:

a'. determine whether rows from a partitioned primary index table are being spooled;

a". determine whether a subsequent operation requires the spooled rows to be ordered in accordance with the first value associated with each row; and

a'". perform steps b through d only if both determinations, a' and a", are true.

39. (original) The computer program of claim 32, wherein the executable instructions cause the computer to:

e. create a file context for each first-merge partition of a subset of the first-merge partitions, each file context storing at least location data for a row in the partition and the first value associated with the row;

f. merge rows from the subset of first-merge partitions into a spool-merge partition in order of the first value associated with each row;

g. repeat steps e and f until the subsets have included all first-merge partitions;

h. bypass steps i through k if the rows from the populated partitions are contained in one partition in order of the first value associated with each row;

i. create a file context for each spool-merge partition of a subset of the spool-merge partitions, each file context storing at least location data for a row in the partition and the first value associated with the row;

j. merge rows from the subset of spool-merge partitions into a new spool-merge partition in order of the first value associated with each row;

k. repeat steps i and j until the rows from the populated partitions are contained in one partition in order of the first value associated with each row.

40. (original) The computer program of claim 39, wherein first-merge partitions and spool-merge partitions are contained in different subtables of a spool.

41. (original) The computer program of claim 39, wherein step j includes merging rows from the subset of spool-merge partitions, each located in a first subtable of a spool, into a new spool-merge partition, located in a second subtable of the spool.

42. (original) The computer program of claim 32, wherein the executable instructions cause the computer to:

e. create a file context for each first-merge partition of a subset of the first-merge partitions, each file context storing at least location data for a row in the partition and the first value associated with the row;

f. merge rows from the subset of first-merge partitions into a spool-merge partition in order of the first value associated with each row;

g. repeat steps e and f until the subsets have included all first-merge partitions;

h. bypass steps i through k if a specified grouping limit is at least equal to the number of spool-merge partitions;

i. create a file context for each spool-merge partition of a subset of the spool-merge partitions, each file context storing at least location data for a row in the partition and the first value associated with the row;

j. merge rows from the subset of spool-merge partitions into a new spool-merge partition in order of the first value associated with each row;

k. repeat steps i and j until the specified grouping limit is at least equal to the number of remaining spool-merge partitions.

43. (original) The computer program of claim 32, wherein the subsets of partitions contain no more than a specified number of populated partitions and the specified number is determined by memory usage.

44. (original) The computer program of claim 32, wherein the executable instructions cause the computer to:

a'. calculate the cost of reorganizing rows from a partitioned database table using the equation cost = (r1 + w) + ((r2 +w) * (ceiling($\log_m$p)-1)), wherein r1 is the cost to read and qualify rows in non-eliminated partitions, w is the cost to write qualifying rows to a spool, r2 is the cost to read the rows in the spool, m is the number of partitions in a subset, p is the number of populated partitions in the table, and ceiling returns an integral argument rounding up.

45. (original) The computer program of claim 32, wherein the reorganization is conducted in response to a query having conditions and the step of merging rows includes eliminating rows that do not satisfy the query conditions.

46. (original) The computer program of claim 32, wherein the first subset of the populated partitions includes all the populated partitions and steps b and c are not repeated.

47. (original) The computer program of claim 32, wherein the first value is the result of a hash function applied to one or more values in one or more columns of the associated row.

48. (original) A method for reorganizing rows from a partitioned database table, the partitioned database table including a plurality of populated partitions, comprising the steps of:

a. organizing rows in each of the populated partitions in accordance with a first value associated with each row;

b. creating a file context for each partition of a subset of the populated partitions, each file context storing at least location data for a row in the partition and the first value associated with the row;

c. merging rows from the subset of partitions into a new populated partition in order of the first value associated with each row, the subset of partitions no longer being counted as populated partitions;

d. repeating steps b through c until no more than a specified number of populated partitions remain.

49. (original) The method of claim 48, wherein steps a through c are performed on rows in a single data-storage facility.

50. (original) The method of claim 48, further comprising the steps of:

a'. determining whether rows from a partitioned primary index table are being spooled;

a". determining whether a subsequent operation requires that the spooled rows be stored in groups ordered in accordance with the first value associated with each row;

a'". determining whether a subsequent operation requires that the spooled rows be stored in a number of groups no more than a specified grouping limit;

a"". performing steps b through d only if the three determinations, a', a", and a'", are true.

51. (original) The method of claim 48, wherein the subsets of partitions contain no more than a specified number of populated partitions and the specified number is determined by memory usage.

52. (original) The method of claim 48, further comprising the step of:

a'. calculating the cost of reorganizing rows from a partitioned database table using the equation cost = (r1 + w) + ((r2 +w) * (ceiling($\log_m p$-$\log_m n$)-1)), wherein r1 is the cost to read and qualify rows in non-eliminated partitions, w is the cost to write qualifying rows to a spool, r2 is the cost to read the rows in the spool, m is the number of partitions in a subset, p is the number of populated partitions in the

table, n is the specified number, and ceiling returns an integral argument rounding up.

53. (original) The method of claim 48, wherein the first value is the result of a hash function applied to one or more values in one or more columns of the associated row.